

|  |  |
| --- | --- |
| Submitted By  Name: Muhammd Mehedi Hasan  Reg: 23201143  Section: C  Subject: CSE 212 | Submitted To  Alif Ruslan  Lecturer  Department of Computer Science and Engineering, [University of Asia Pacific](https://uap-bd.edu/) |

**Module 1: EMPLOYEE Table**

**Create Table**

CREATE TABLE EMPLOYEE (

empId INTEGER PRIMARY KEY,

name TEXT NOT NULL,

dept TEXT NOT NULL

);

🔹 *Creates the EMPLOYEE table with empId, name, and dept attributes.*

**Insert Data**

INSERT INTO EMPLOYEE VALUES (0001, 'Clark', 'Sales');

INSERT INTO EMPLOYEE VALUES (0002, 'Dave', 'Accounting');

...

INSERT INTO EMPLOYEE VALUES (00012, 'Kelvin', 'Sales');

🔹 *Inserts employee records into the EMPLOYEE table.*

**Select Queries**

SELECT name FROM EMPLOYEE;

SELECT dept FROM EMPLOYEE;

SELECT \* FROM EMPLOYEE WHERE dept = 'Accounting' OR dept = 'Sales';

🔹 *Retrieves names, departments, and employees from specific departments.*

**Result (brief):** Shows employee names, departments, and a filtered list for Accounting & Sales.

**Delete & Update**

-- Delete Sales & Marketing employees

DELETE FROM EMPLOYEE WHERE dept = 'Sales' OR dept = 'Marketing';

-- Update a record

UPDATE EMPLOYEE SET name = 'Alfred Schmidt', dept = 'Marketing' WHERE empId = 0001;

🔹 *Removes certain employees and updates details of empId=1.*

**Result (brief):** Sales/Marketing staff deleted, empId=1 updated to Alfred Schmidt.

**Module 2: Department Table**

**Create & Insert**

CREATE TABLE Department (

studentId INT PRIMARY KEY,

name TEXT NOT NULL,

dept TEXT NOT NULL

);

INSERT INTO Department VALUES (001, 'Ahad', 'ENG');

...

INSERT INTO Department VALUES (0010, 'Adib', 'CIVIL');

🔹 *Creates Department table and inserts student data.*

**Update with CASE**

UPDATE Department SET studentId = CASE

WHEN studentId = 1 THEN 2001

...

WHEN studentId = 10 THEN 2010

ELSE studentId

END;

🔹 *Updates studentId values by mapping them to new IDs.*

**Update EMPLOYEE with CASE**

UPDATE EMPLOYEE SET name = CASE

WHEN empId > 6 THEN 'Okay'

ELSE 'Not Okay'

END;

🔹 *Changes names conditionally based on empId.*

**Result (brief):** EMPLOYEE names updated to either "Okay" or "Not Okay".

**Module 3: ClassRoom Table**

**Create & Insert**

CREATE TABLE ClassRoom (

building INTEGER NOT NULL,

room\_number TEXT NOT NULL,

capacity INTEGER

);

INSERT INTO ClassRoom VALUES (1, '101', 42);

...

INSERT INTO ClassRoom VALUES (6, '604', 42);

🔹 *Defines classroom details with building, room number, and capacity.*

**Select & Delete**

SELECT \* FROM ClassRoom;

DELETE FROM ClassRoom WHERE building = 4;

SELECT \* FROM ClassRoom WHERE building < 5;

🔹 *Shows all records, deletes building=4, then selects buildings less than 5.*

**Filter with OR/AND**

SELECT \* FROM ClassRoom WHERE building = 2 OR capacity = 42;

SELECT \* FROM ClassRoom WHERE building = 2 AND capacity = 52;

🔹 *Demonstrates conditional filtering using OR/AND.*

**Schema Modification**

-- Add column

ALTER TABLE ClassRoom ADD Email VARCHAR(255) NOT NULL;

-- Drop column

ALTER TABLE ClassRoom DROP COLUMN capacity;

-- Rename column

ALTER TABLE ClassRoom RENAME COLUMN building TO BUILDING\_NO;

-- Update new column

UPDATE ClassRoom SET Email = '123abc@gmail.com' WHERE BUILDING\_NO > 0;

🔹 *Adds, deletes, renames columns, and updates new Email field.*

**Result (brief):** ClassRoom table schema modified, Email column added and filled.

**Final Result Discussion (Very Brief)**

* **EMPLOYEE Table**: Successfully demonstrates insert, select, delete, and conditional update queries.
* **Department Table**: Showcases update with CASE, renaming IDs correctly.
* **ClassRoom Table**: Covers schema modification operations (ALTER, DROP, RENAME) and conditional queries.
* **Overall**: The script covers all key SQL operations—**CRUD + Schema Alteration**.